What is Claimed:

1. A method for logging while updating a B-link tree via a plurality of data transactions, whereby a current state of the data structure is recovered by re-performing data transactions represented by the logging, comprising:

generating at least one log entry corresponding to a data transaction of the plurality of data transactions, the data transaction to be carried out on said B-link tree; and

storing said at least one log entry into a log.

2. A method according to claim 1, further including periodically truncating the log.

3. A method according to claim 1, wherein said at least one log entry includes at least one of (A) at least one entry from an allocation layer and (B) at least one entry from a B-link tree layer.

4. A method according to claim 1, further including discarding said at least one log entry from the log when the data transaction has been carried out on said B-link tree.

5. A method according to claim 1, wherein said storing includes storing said at least one log entry into the log before the data transaction is carried out on said B-link tree.

6. A method according to claim 1, further including caching data of said data transaction before said data transaction is carried out on said B-link tree.

7. A method according to claim 1, further including storing said at least one log entry in an intermediate memory previous to storing said at least one log entry in the log.

8. A method according to claim 7, wherein said at least one log entry is moved from intermediate memory to the log after the data transaction commits.

9. A method according to claim 1, further including maintaining a log sequence number with each of said at least one log entry, uniquely identifying said at least one log entry.

10. A computer readable medium comprising computer executable instructions for performing the method of claim 1.

11. A modulated data signal carrying computer executable instructions for performing the method of claim 1.

12. A method for logging while updating a B-link tree via a plurality of data transactions, whereby a current state of the data structure is recovered by re-performing data transactions represented by the logging, comprising:

generating at least one log entry corresponding to a data transaction of the plurality of data transactions, the data transaction to be carried out on said B-link tree;

storing said at least one log entry into a finite log;

periodically flushing data corresponding to data transactions represented by the finite log to persistent storage; and

truncating said finite log in coordination with said flushing.

13. A method according to claim 12, wherein said at least one log entry includes at least one of (A) at least one entry from an allocation layer and (B) at least one entry from a B-link tree layer.

14. A method according to claim 12, further including discarding said at least one log entry from the finite log when the data transaction has been carried out on said B-link tree.

15. A method according to claim 12, wherein said storing includes storing said at least one log entry into the finite log before the data transaction is carried out on said B-link tree.

16. A method according to claim 1, further including caching data of said data transaction before said data transaction is carried out on said B-link tree.

17. A method according to claim 12, further including storing said at least one log entry in an intermediate memory previous to storing said at least one log entry in the finite log.

18. A method according to claim 17, wherein said at least one log entry is moved from intermediate memory to the finite log after the data transaction commits.

19. A method according to claim 12, further including maintaining a log sequence number with each of said at least one log entry, uniquely identifying said at least one log entry.

20. A computer readable medium comprising computer executable instructions for performing the method of claim 12.

21. A modulated data signal carrying computer executable instructions for performing the method of claim 12.

22. A method for logging while updating a data structure via a plurality of data transactions, whereby a current state of the data structure is recovered by re-performing data transactions represented by the logging, comprising:

replicating updates to the data structure to a first server location and a second server location;

generating at least one log entry corresponding to a data transaction of the plurality of data transactions, the data transaction to be carried out on said data structure; and

maintaining a single log, where the log is partitioned into an upper layer and an allocation layer, at each of said first and second server locations, wherein the single log includes log entries from both the upper layer and allocation layer.

23. A method according to claim 22, further including recovering the data structure after a failure by performing parallel recovery operations by each of said first and second server locations.

24. A method according to claim 22, wherein said data structure is a B-link tree.

25. A method according to claim 24, wherein the upper layer is a B-link tree layer that handles B-link tree operations.

26. A method according to claim 22, wherein the allocater layer handles at least one of (A) an allocate disk space operation, (B) a deallocate disk space operation, (C) a read from the allocated disk space operation and (D) a write to the allocated disk space operation.

27. A computer readable medium comprising computer executable instructions for performing the method of claim 22.

28. A modulated data signal carrying computer executable instructions for performing the method of claim 22.

29. A server for maintaining a log while updating a B-link tree via a plurality of data transactions, whereby a state of the data structure is recovered by re-performing data transactions represented by the logging, comprising:

a logging object that generates at least one log entry corresponding to a data transaction of the plurality of data transactions, the data transaction to be carried out on said B-link tree; and

a storage log for storing said at least one log entry.

30. A server according to claim 29, wherein the finite storage log including said at least one log entry is periodically truncated.

31. A server according to claim 29, further comprising:

an allocation layer object for said B-link tree; and

a B-link tree layer object,

wherein said at least one log entry includes at least one of (A) at least one entry from the allocation layer object and (B) at least one entry from the B-link tree layer object.

32. A server according to claim 29, wherein said at least one log entry is discarded from the storage log when the data transaction has been carried out on said B-link tree.

33. A server according to claim 29, wherein said at least one log entry is stored in the storage log before the data transaction is carried out on said B-link tree.

34. A server according to claim 29, wherein data of said data transaction is cached in a cache memory before said data transaction is carried out on said B-link tree.

35. A server according to claim 29, further including storing said at least one log entry in an intermediate memory previous to storing said at least one log entry in the storage log.

36. A server according to claim 35, wherein said at least one log entry is moved from intermediate memory to the storage log after the data transaction commits.

37. A server according to claim 29, wherein said logging object generates a log sequence number with each of said at least one log entry, uniquely identifying said at least one log entry.

38. A server for logging while updating a B-link tree via a plurality of data transactions, whereby a state of the data structure is recovered by re-performing data transactions represented by the logging, comprising:

a first object for generating at least one log entry corresponding to a data transaction of the plurality of data transactions, the data transaction to be carried out on said B-link tree;

a finite storage log for storing said at least one log entry;

a second object for periodically flushing data corresponding to data transactions represented by the at least one log entry in the finite storage log to persistent storage;

a third object for truncating said finite storage log in coordination with the operation of the flushing of the second object.

39. A server according to claim 38, further including:

an allocation layer and a B-link tree layer, wherein said at least one log entry includes at least one (A) at least one entry from the allocation layer and (B) at least one entry from the B-link tree layer.

40. A server according to claim 38, wherein said at least one log entry is discarded from the finite storage log when the data transaction has been carried out on said B-link tree.

41. A server according to claim 38, wherein said at least one log entry is stored in the finite storage log before the data transaction is carried out on said B-link tree.

42. A server according to claim 38, wherein data of said data transaction is cached in a cache memory before said data transaction is carried out on said B-link tree.

43. A server according to claim 38, further including storing said at least one log entry in an intermediate memory previous to storing said at least one log entry in the finite storage log.

44. A server according to claim 43, wherein said at least one log entry is moved from intermediate memory to the finite storage log after the data transaction commits.

45. A server according to claim 38, wherein said first object generates a log sequence number with each of said at least one log entry, uniquely identifying said at least one log entry.

46. A server for logging while updating a data structure via a plurality of data transactions, whereby a state of the data structure is recovered by re-performing data transactions represented by the logging, comprising:

a replication object that replicates updates to the data structure to a first server location and a second server location;

a logging object that generates at least one log entry corresponding to a data transaction of the plurality of data transactions, the data transaction to be carried out on said data structure; and

a storage element within which a single log is maintained, wherein the single log is partitioned into an upper layer and an allocation layer, at each of said first and second server locations, and wherein the single log includes log entries from both the upper layer and allocation layer.

47. A server according to claim 46, wherein the data structure is recovered after a failure via parallel recovery operations by each of said first and second server locations.

48. A server according to claim 46, wherein said data structure is a B-link tree.

49. A server according to claim 48, wherein the upper layer is a B-link tree layer that handles B-link tree operations.

50. A server according to claim 46, wherein the allocater layer handles at least one of (A) an allocate disk space operation, (B) a deallocate disk space operation, (C) a read from the allocated disk space operation and (D) a write to the allocated disk space operation.

51. A computing device for logging while updating a B-link tree via a plurality of data transactions, whereby a current state of the data structure is recovered by re-performing data transactions represented by the logging, comprising:

means for generating at least one log entry corresponding to a data transaction of the plurality of data transactions, the data transaction to be carried out on said B-link tree; and

means for storing said at least one log entry into a log.

52. A computing device according to claim 51, further including means for truncating the log periodically.

53. A computing device for logging while updating a B-link tree via a plurality of data transactions, whereby a current state of the data structure is recovered by re-performing data transactions represented by the logging, comprising:

means for generating at least one log entry corresponding to a data transaction of the plurality of data transactions, the data transaction to be carried out on said B-tree;

means for storing said at least one log entry into a finite log;

means for periodically flushing data corresponding to data transactions represented by the finite log to persistent storage; and

means for truncating said finite log in coordination with said means for periodically flushing.

54. A computing device according to claim 53, further including means for discarding said at least one log entry from the finite log when the data transaction has been carried out on said B-link tree.

55. A computing device for logging while updating a data structure via a plurality of data transactions, whereby a current state of the data structure is recovered by re-performing data transactions represented by the logging, comprising:

means for replicating updates to the data structure to a first server location and a second server location;

means for generating at least one log entry corresponding to a data transaction of the plurality of data transactions, the data transaction to be carried out on said data structure; and

means for maintaining a single log, where the log is partitioned into an upper layer and an allocation layer, at each of said first and second server locations, wherein the single log includes log entries from both the upper layer and allocation layer.

56. A computing device according to claim 55, wherein said data structure is recoverable after a failure by performing parallel recovery operations by each of said first and second server locations.